1. JPA algemeen.

1.1. Eén domeinklasse en drie objecten.

We maken één domeinklasse **Docent** aan. We instantiëren drie docent objecten en zorgen ervoor dat die in de databank schoolDB in de tabel Docenten worden bijgehouden.
Van een docent houden we de **docentNr** (int), **voornaam** (String) , **familienaam** (String) en **wedde** (Bigdecimal) bij.

Maak een Java Application project aan.
Voeg de EclipseLink (JPA2.1) library toe aan het project. (project properties/libraries/Add Library).
Maak de **Docent** klasse aan in het domein.
Voeg de gepaste annotations toe.
We voorzien als primary key in de database een autonummering. In de docentklasse voorzien we hiervoor een attribuut met naam **id**.
id is de primary key en is autonumber in de database.

```
@GeneratedValue( strategy = GenerationType.IDENTITY)
```

We willen dat het attribuut **docentnr** met een veld **PERSONEELSNR** wordt gemapped.

Voorzie ook setters, getters, constructor en toString methoden. Zorg ervoor dat twee docentobjecten gelijk zijn als hun docentnr gelijk is.

Voorzie ook een (protected) default constructor, verplicht voor een entity klasse.

Maak een DB aan:

```
Services:
    Databases:
        Java DB: rechts klik --> Create Java DB Database
            Databas eName: schoolDB
            User Name:  root
            Password: rootp
```

Maak een Persistence Unit. (selecteer het project: File/new File/Persistence/Persitence Unit)

```
geef PU naam: school
geef DB connectie:
    database connection: jdbc:derby://localhost:1527/schoolDB
    table generation strategy: create
```

-->persistence.xml wordt aangemaakt in map META-INF

```
voeg de entity classes toe: domein.Docent   --> Add Class --> domein.Docent
```

Maak een startup om de objecten aan te maken en persistent te maken.

```
Maak een nieuw package main aan.
Maak een java klasse MAINoef1 met de main methode in.
In de main methode:
    Maak drie Docent objecten aan.
```

Maak een EntityManager aan.

gebruik hiervoor de EntityManagerFactory.

Een EntityManagerFactory instance maken vraagt veel tijd.
– Je maakt zo'n instance één keer per applicatie.
– Je houdt hem best bij via een singleton utility class.

Maak een package util aan. Plaats daar het singleton JPAUtil dat een EntityManagerFactory object herbergt. Je dient voor de creatie hiervan de persistence unit naam door te geven: school.

Met de createEntityManager van de factory krijg je dan een instance.
Start een transactie.
Persist de drie Docent objecten.
Commit de transactie.
Sluit de entitymanager
Sluit de factory.

Voor je de applicatie kan runnen dien je ook nog de derbyclient.jar toe te voegen aan het project, nodig voor de database drivers.

--> in je project --> libraries -> klik rechts --> add jar/folder --> derbyclient.jar

De jar is in de mappen van je JDK (vb: C:\Program Files\Java\jdk1.8.0_131\db\lib).

Run de applicatie en bekijk het resultaat in de DB.

Om het resultaat te bekijken:

Services:

Klik rechts op de connectie vd DB -> Connect -> ROOT -> Tables -> DOCENTEN
Je ziet het ontwerp van de aangemaakte tabel.
Klik rechts -> view data.
Je ziet de aangemaakte records.

1.2. Een object wijzigen.

Maak een java klasse MAINoef2 met de methode main in. Voorzie de nodige stappen om een nieuwe transactie uit te voeren. MAINoef1 is uitgevoerd. In de database zijn drie records.
Je haalt het docentobject met id 2 op en verhoogt zijn wedde met 200 euro.
Je kan selectief één object ophalen via de entitymanager door de find bewerking:

-->em.find(Docent.class, 2L)

Het eerste argument bepaalt de klasse van het object dat gezocht wordt en het tweede argument is een voorkomen van de primaire sleutel. Indien geen record gevonden wordt, krijgen we een null referentie.

Bekijk het resultaat in de DB.